Kripke's paradox and the Church-Turing thesis

Published in *Synthese* (2008) 160: 285–295.
doi:10.1007/s11229-006-9120-2
mark.sprevak@ed.ac.uk

Kripke's paradox and the Church–Turing thesis

Mark Sprevak
University of Edinburgh

10 October 2006

Kripke (1982) presents a rule-following paradox in terms of what we meant by our past use of 'plus', but the same paradox can be applied to any other term in natural language. Many responses to the paradox concentrate on fixing determinate meaning for 'plus', or for a small class of other natural language terms. This raises a problem: how can these particular responses be generalised to the whole of natural language? In this paper, I propose a solution. I argue that if natural language is computable in a sense defined below, and the Church–Turing thesis is accepted, then this auxiliary problem can be solved.

1 Introduction

I do not aim to solve Kripke's rule-following problem in this paper. I address a related problem that should be a source of worry for respondents to Kripke. This problem is as follows. Kripke's challenge about meaning is general: it can be applied to any term in natural language. However, many responses to his challenge are specific: they aim to fix meaning for particular terms or classes of terms. The problem that I address is how to extend these particular solutions to the entirety of natural language. I call this the 'auxiliary problem'. The auxiliary problem, although not Kripke's rule-following problem, is important nonetheless. Even if one managed to solve the rule-following problem for a proper subset of natural language—an incredibly hard task—the auxiliary problem would remain. The auxiliary problem has received little or no attention in the literature. In this paper, I describe the auxiliary problem, and I propose a solution. I argue that, provided natural language is computable in a certain sense defined below, a result in computation theory, the Church–Turing thesis, provides a solution.

Kripke's challenge can be summarised as follows. Kripke's sceptic asks what fact determined that you meant *plus* rather than *quus* in your past use of the term 'plus', where *quus*

is a function just like the addition function for all sums below those you have already computed, but diverges for higher numbers. The same question, suitably modified, can be asked about any term in natural language. The sceptic's challenge has generated a huge number of responses.[1] Some of the leading responses attempt to state a fact that fixes meaning. Such 'straight' replies tend to restrict attention to a few terms, or to certain kinds of terms. Solving the problem, even for these limited cases, would be a major achievement. However, even if success is granted, the problem remains of generalising the solution. If this cannot be done, then Kripke's sceptic would be free to press her question for other terms.

Maddy (1984) and McGinn (1984) argue that the sceptic's challenge can be answered for proper names (e.g. 'Cicero') and natural kind terms (e.g. 'water') by Kripke's own causal theory of reference. However, is not clear how the causal theory of reference could fix determinate meaning for terms that are neither proper names nor natural kind terms (e.g. 'friend', 'scientist'). Even if Maddy and McGinn are correct about proper names and natural kind terms, it is not clear how their reply could be a general solution. Millikan (1990) gives a different reply to the sceptic. She argues that terms that are associated in salient ways with our evolutionary history (e.g. 'milk', 'danger') have their meaning fixed by facts about that history. Again, even if correct, this has limited scope as a solution. Not all terms have distinctive associations with our evolutionary past. It is not clear, for example, how Millikan's account could fix the meaning of terms associated with recent concepts such as 'herbaceous border' or 'wicket keeper'. Fodor (1990) advocates yet another response to the sceptic. He argues that the meanings of predicates are fixed by asymmetric nomic relations that obtain between the tokening of a predicate, such as '…is a cow', and the property that it names, *being a cow*. Again, whatever the merits of this solution, it is unlikely to succeed as a general account. For one thing, it requires a property to exist for every meaningful predicate: a meaning-constituting asymmetric nomic relation cannot obtain unless both predicate tokening and corresponding property exist. However, it is far from clear which properties exist, and whether there are enough properties for each meaningful predicate. Even if Fodor is correct about some terms, there may be other terms without corresponding properties for which the sceptic can still press her question.

The auxiliary problem concerns how to extend particular solutions, such as those of Maddy and McGinn, Millikan, or Fodor, to the entirety of natural language. One simple-minded approach would be to take the conjunction of all existing solutions, so that each fixes the meaning of its own specialised terms. Sadly, in addition to being inelegant, this approach is unlikely to succeed. The existing solutions presuppose different, and incompatible, views of the underlying metaphysics. It would be inconsistent to take their conjunction. Furthermore, there is no guarantee that a conjunction of existing approaches would be able to cover the required ground: it is not obvious that even a conjunction would exhaust natural language. In this paper, I argue for an alternative approach. Kripke himself, in the context of a different discussion, indicates the way to proceed.

1. For example, see Miller and Wright (2002) and references.

2 The algorithm reply

One possible reply to the sceptic that Kripke considers is the 'algorithm reply'. The algorithm reply asserts that you meant *plus* rather than *quus* in your past use of 'plus' if you followed an addition algorithm when you applied 'plus' rather than a quaddition algorithm. Kripke gives an example of an addition algorithm:

> Take a huge bunch of marbles. First count out x marbles in one heap. Then count out y marbles in another. Put the two heaps together and count out the number of marbles in the union thus formed. The result is $x + y$.
>
> (Kripke 1982, p. 15)

The algorithm reply claims that if you followed these instructions, rather than those of a quus-like algorithm, then you succeeded in meaning *plus* rather than *quus* by 'plus'. However, as Kripke points out, this won't do as an answer to the sceptic, since the sceptic can repeat her challenge on the terms used by the algorithm. She can ask: what fact determined in your past use of the term 'count' that you meant *count* rather than *quount*—'where to 'quount' a heap is to count it in the ordinary sense, unless the heap was formed as the union of two heaps, one of which has 57 or more items, in which case one must automatically give the answer '5'' (p. 16). If one replies that one meant *count* rather than *quount* because one followed a *counting* algorithm rather than a *quounting* algorithm, then the sceptic can repeat her challenge on the terms of that algorithm, and so on. Kripke is undoubtably right that the algorithm reply, by itself, fails miserably as an answer to the sceptic. However, I wish to suggest that the algorithm reply, with a little help from the Church–Turing thesis, can play a different role: it can solve the auxiliary problem.

The auxiliary problem is the problem of extending answers to the sceptic for particular terms to the whole of natural language. Call the terms for which we already have an answer to the sceptic (those terms for which we already have a particular solution), the 'basic terms'. Call the other terms, those for which we do not yet have an answer to the sceptic, the 'non-basic terms'. In order for the algorithm reply to work, the meaning of the non-basic terms has to be fixable using only basic terms. At first sight, this looks unlikely. Natural language is incredibly heterogeneous. The claim that it can be reduced to combinations of a few basic terms looks implausible.

Before considering why the success of the algorithm reply seems unlikely, let us first consider the reply in more detail. The algorithm reply generalises a particular solution to sceptic to the entirety of natural language. In order to do this, it requires: (1) that we follow algorithms that give the conditions of correct application of all non-basic terms; (2) that those algorithms use only basic terms. If these two conditions are met, then the algorithm reply works as follows: in your past use of a non-basic term $\ulcorner x \urcorner$ you meant x by it just in case you followed an appropriate algorithm when you applied the term. If you followed an appropriate $\ulcorner x \urcorner$-algorithm when you applied $\ulcorner x \urcorner$, then you meant x by $\ulcorner x \urcorner$; if you did not follow an appropriate $\ulcorner x \urcorner$-algorithm, then you did not mean x by $\ulcorner x \urcorner$.

It is important to see that if (1) and (2) are met, then the algorithm reply provides a watertight answer to the auxiliary problem. In the 'plus' example above, the sceptic was able to raise her question by questioning the meanings of the component terms of the

'plus' algorithm. That strategy is now blocked. By hypothesis, all of the terms used by the algorithms of the algorithm reply are basic. Therefore, the sceptic's problem cannot be raised again for those terms. Perhaps a different strategy is open to the sceptic. Perhaps she can question the determinacy of the notion of 'following an algorithm'. However, this strategy is blocked too. What is involved in following a particular algorithm is fully spelled out by the terms of that algorithm. If those terms have determinate meaning, then it is determinate what is involved in following that algorithm. The sceptic cannot, for example, question whether one is really followed an algorithm *A* or a divergent algorithm *quA* when applying a term ⌜*x*⌝. The only way that one could have followed a divergent algorithm *quA* would be if algorithm *A* had different terms, or terms with different meanings. But by hypothesis, algorithm *A* has fixed terms with fixed meanings. So if one follows an algorithm with the terms of algorithm *A*, there can be no question of one following a different algorithm *quA*. Algorithms, if available at all, provide a watertight answer to the auxiliary problem.

There are three reasons why one might doubt that the algorithm reply can work.

First, one might doubt that one has the right set of basic terms. The algorithm reply requires that the conditions of correct application of every non-basic term can be defined using only basic terms. How can one be sure that one's collection of basic terms is up to the job? Short of the Herculean task of actually providing all the algorithms, there seems no reason for thinking that one's collection is adequate. Which terms one counts as basic will depend on the solution one favours to Kripke's sceptical problem. If one follows Maddy and McGinn, then the basic terms will be proper names and natural kind terms. If one follows Millikan, then the basic terms will be evolutionarily salient terms. If one follows Fodor, then the basic terms will be those whose tokenings stand in asymmetric nomic relations with their corresponding properties. What reason do we have for thinking that any of these terms can specify the conditions of correct application of all other terms in natural language?

The second worry is that natural language may not be reducible to basic terms at all. Natural language could turn out to be holistic rather than reducible. Natural language is holistic just in case there is *no single small* subset of basic terms from which all other terms can have their conditions of correct application defined. If natural language is holistic, then although the conditions of correct application of some words can be given in terms of others, no systematic reduction to a single basic set of terms is possible. In other words, there is no privileged set of terms whose determinacy, if fixed, would fix the determinacy of the rest of language too. Natural language would not 'bottom out' in a single small basic vocabulary, instead it would hang together in a holistic web of definitions. If natural language is holistic, then the algorithm reply is doomed: *whatever* basic set of terms one chooses—short of the limit of choosing every term in natural language—the algorithm reply cannot extend that solution to other terms. In order for the algorithm reply to work, natural language must be reducible. Natural language is reducible just in case *there is* a small subset of basic terms from which all other terms can have their conditions of correct application defined. The worry is that we seem to have no reason to think that natural language is reducible, or reducible to any significant degree. Indeed, intuitions seem to favour a contrary nature: a glance at a dictionary seems to show a holistic structure, rather than reduction to a single basic vocabulary.

The third worry is that providing an algorithm for how to apply a term is tantamount to giving a definition of that term, and the existence of definitions is, in general, suspect. Doubts about the existence of definitions come from a number of sources, including Wittgenstein's discussion of the difficulty of defining 'game', Quine's argument against the existence of analytic truths, and cognitive science work, such as that of Rosch (1973), on the non-definitional nature of our concepts. The idea that terms generally have definitions looks, on many grounds, to be suspect. Therefore, the sceptic's choice of a term that *does* have a definition, 'plus', may give the misleading impression that the algorithm reply can do more work than it can.

All of these concerns about the algorithm reply can be answered. With a small elaboration, the algorithm reply can be shown to be an adequate solution to the auxiliary problem.

3 The computability assumption

Before discussing the solution, the idea needs to be introduced that following a rule can be understood as computing a function. This is not intended to be controversial.[2] Functions are mappings from one set to another. Rules are instructions on how to act: they tell one, in this situation perform this action. It is possible to think of a rule as a function mapping a set of situations to a set of actions, and the process of following a rule as the process of computing that function.

There are at least two senses of rule, and two senses of function. Following the terminology of Church (1941), the term 'function' can refer to a function-in-extension or a function-in-intension. A *function-in-extension* is the set of input/output pairs associated with the function. For example, the function $f(x) = x^2$ where x is a positive integer, considered as a function-in-extension, is the set of input/output pairs $\{(1,1),(2,4),(3,9),\ldots\}$. A *function-in-intension* is a method for computing the set of input/output pairs. For example, the function $f(x) = x^2$ where x is a positive integer, considered as a function-in-intension could be the method 'Multiply x by itself—take the result as $f(x)$'.[3] Functions-in-extension are easily individuated: two functions-in-extension are the same just in case the two sets of I/O pairs are the same. The individuation conditions of functions-in-intension, outside a formalism like Church's λ-calculus, are less clear. However, it is generally true that functions-in-intension individuate more finely than functions-in-extension.[4]

The term 'rule' is similarly ambiguous. A rule could be a prescribed set of situation–action pairs; a rule could prescribe: in *this* situation perform *this* action. For example, the rule 'Always wash your hands before you eat', could be understood as the prescribed set of situation–action pairs: { (occasion$_1$ of eating, wash hands before), (occasion$_2$ of eating, wash hands before), … }. Alternatively, the rule 'Always wash your hands before you eat' could be understood as *a way of achieving* those situation–action pairs: for example, 'Before eating, walk to the bathroom, turn on the tap, wet your hands, pick up the soap, etc.'. The equivalence claim above between functions and rules is intended as a dual equivalence claim: an equivalence between functions-in-extension and rules understood

2. Ginet (1992) defends a similar equivalence of functions and rules.
3. In Church's λ-notation: $f(x) = \lambda x.xx$.
4. Church (1941), pp. 2–3.

extensionally, and an equivalence between functions-in-intension and rules understood intensionally. I will distinguish the two senses by referring to functions-in-extension and rules understood extensionally as 'functions', and functions-in-intension and rules understood intensionally as 'algorithms' or 'rules'.

In order for Kripke's sceptic to raise her challenge, the application conditions of natural language terms must be governed by rules. It is this assumption that enables the sceptic to transfer her concerns from rules in general to the application of natural language terms. In what follows, I take it for granted that the assumption that application of natural language terms is rule-governed is agreed on by all sides. The algorithm reply to the auxiliary problem adds an extra requirement to this assumption: that those rules be, in the sense defined below, *computable*.

This requirement can be phrased in many ways: as the requirement that the rules governing the application of natural language terms be effective procedures, as the requirement that the decision as to whether a natural language term correctly applies is determinable by finite means, as the requirement that a computer could, in principle, decide in finite time whether a given natural language term correctly applies in a certain case, or as the requirement that a computer could, in principle, reproduce human linguistic ability. Four things should be made clear about the computability assumption.

First, the computability assumption does not beg the question against the sceptic. Even if the computability assumption is correct, the sceptic is free to run her argument. The computability assumption entails that a rule for the correct application of a natural language term can given in terms that a computer can execute, but it does not entail that those terms themselves have determinate meaning. For example, the computability assumption entails that a rule for the correct application of a natural language term can be given in terms of the basic operations of a Turing machine, such as 'scan symbol', 'write symbol', 'erase symbol', and so on. But the computability assumption does not entail that these operations themselves have determinate meaning, i.e. that 'scan symbol' means *scan symbol* and not *squan symbol* (where to squan a symbol is the same as to scan a symbol unless that symbol is '57' in which case the answer is '5'). If one wishes to argue that 'scan symbol' means *scan symbol*, one needs a separate argument to that effect.

Second, many respondents to Kripke's sceptic are either sympathetic or already committed to the computability assumption. For instance, nearly all advocates of the computational theory of mind (CTM) are committed to it. The computability assumption cuts across the main disagreements within the CTM, such as whether the mind has a classical or a connectionist architecture. No matter what kind of computer the mind is, if that computer is responsible for our linguistic abilities, then the rules that govern those abilities must, in the sense above, be computable.

Third, the computability assumption entails that a rule that a computer is capable of following can be given for the application of each natural language term, but it does not entail that there must only be one such rule for each term. The same function can be computed in many different ways. A natural language term may therefore have more than one algorithm associated with its conditions of correct application.

Finally, over and above the commitments of philosophers to the computability assumption, there seem to be good reasons for taking that assumption on board, at least pending good

reasons otherwise. First, computable methods have massive expressive power. There is so much that can be done with computable functions that it seems strange to lose hope before one starts. Many cases of apparent incomputable behaviour can be dealt with without violating the core of the computability assumption.[5] Second, one has to opt out of mainstream intuitions if one denies the computability assumption. Denying the computability assumption entails that a computer, even in the form of a humanoid robot, could never master natural language. This seems like a controversial *a priori* commitment to make.[6]

None of these reasons is conclusive. I do not know of any argument strong enough to show that the rules governing natural language must be computable. All that I wish to claim is that the computability assumption is *prima facie* plausible: it should be accepted pending good reasons otherwise. Given the magnitude of assumptions that get made in solutions to Kripke's sceptic, the computability assumption is a fairly mild addition that yields a healthy pay-off.

4 The pay-off

Let us return to the reasons why the algorithm reply was suspected to be unsatisfactory, and consider those reasons in reverse order.

First, there was a worry that providing an algorithm for how to apply a term is tantamount to providing a definition of that term. This is false. Algorithms are not intended to be definitions, they are intended to provide extensional equivalence. Extensional equivalence is a much weaker condition than synonymy. There is no reason to think that doubts about the existence of definitions should carry over to doubts about the existence of algorithms for applying terms. Indeed, the computability assumption entails that they do not. If the computability assumption is true, then there must be computable rules governing the correct application of natural language terms. Hence, there must exist algorithms for applying those terms. These algorithms fix the correct application conditions of those terms, and this is exactly what Kripke's sceptic questions.

5. For example, consider terms such as '…is a non-halting Turing machine', whose correct application is in principle not decidable by a computer. There are at least three ways of dealing with such cases. One is to treat them as non-computable composites made up of components, such as negation, 'halting', and 'Turing machine', that are governed by computable rules. Another is to treat them as promissory notes on which we cannot actually deliver over the full infinite domain, but to which we can provide a finite approximation. Finally, one could treat such expressions as having a different semantic content from the descriptive role that they appear to have, e.g. give them an expressivist treatment.

6. Some philosophers claim to demonstrate *a priori* that a computer cannot understand natural language (e.g. Searle (1980), Dreyfus (1992), Haugeland (1981)). However, the computability assumption is strictly speaking weaker than the claims that these philosophers attack. First, as discussed in Section 5, the computability assumption does not entail that computations take place inside individual human heads. Second, the computability assumption is true provided a simulation of a human (Searle), a connectionist architecture (Dreyfus), or a suitably human-like robot (Haugeland) can correctly apply natural language terms. Third, the computability assumption only concerns correctness conditions, it does not concern all aspects of language understanding, or indeed any other mental process. The computability assumption may be false, but it is not obviously incompatible with the principal arguments against the computational view of the mind.

The second worry was that natural language may be holistic rather than reducible. If natural language is holistic, then the algorithm reply cannot solve the auxiliary problem: no matter which set of terms one takes as basic, there is no small subset from which the application conditions of all other terms can be defined. However, if the computability assumption is true and the Church–Turing thesis is accepted, then natural language can be shown to be reducible.

The computability assumption entails that, *for each term*, there are finite means in terms of which its correct application can be specified. However, the computability assumption does not entail that *the same* finite means—the same set of basic terms—work in each case. For example, it may be that a term ⌜a⌝ has an algorithm governing its correct application that uses the set of basic terms {⌜x⌝, ⌜y⌝, ⌜z⌝}, and that another term ⌜b⌝ has an algorithm governing its correct application that uses a different set of basic terms {⌜u⌝, ⌜v⌝, ⌜w⌝}. It may be that each non-basic term in natural language requires a different set of basic terms. Even if computable algorithms determine the conditions of correct application of terms, that does not mean that the total number of terms used by those algorithms is any fewer than the total number of terms in natural language. It is compatible with the computability assumption that language be holistic. Fortunately, there is another result that shows that language is reducible under these conditions: the Church–Turing thesis.

The Church–Turing thesis holds that if a function—an input/output pattern—is computable at all, then it is computable using algorithms that use *only a small number of basic terms*. Computers can produce an incredible variety of I/O behaviour, including, if the computability assumption is true, the linguistic I/O behaviour of humans. One might think that such a variety of I/O behaviour has to be reflected in a corresponding variety of basic predicates. The Church–Turing thesis says that this is not the case. If a rule can be specified using *some finite means or other* (which the computability assumption asserts), then it can be specified using one of any number of small sets of basic predicates. A small set of basic terms is enough to specify algorithms for producing any computable behaviour. Therefore, if the Church–Turing thesis is true, language is reducible a small set of basic terms.

The final worry was whether one's set of basic terms is up to the job of defining the conditions of correct application for all other terms. Depending on which response to the sceptic one favours—Maddy's and McGinn's, Millikan's, or Fodor's—one will end up with a different set of basic terms. How can one be sure that one's set of basic terms is adequate to define the conditions of correct application of all other terms? If the computability assumption and the Church–Turing thesis are correct, then this question can be given at least a provisional answer: a set of basic terms is adequate just in case that set defines the architecture of a universal computing machine.

A universal computing machine is a machine that can, in principle, compute any computable function. If natural language is computable at all (which it is by the computability assumption), then it is computable by any universal computing machine. The best known universal computing machine is the universal Turing machine, but there are plenty of other examples: universal register machines, universal Post machines, universal automata in Conway's game of life, and electronic PCs with unbounded memory. These machines differ in the algorithms that they run, but they share the characteristic of being capable of reproducing the input/output pattern of any other computer. The algorithms that a

universal machine runs consist in finite combinations of a finite number of basic instructions. For a Turing machine, the basic instructions are 'scan symbol', 'erase symbol', 'move head left', and so on; for a register machine the basic instructions are 'increment register', 'decrement register', 'branch if zero', and so on. The basic instructions of a machine are the basic terms of that machine's algorithms. If those basic instructions have a fixed meaning—a meaning immune to sceptical reinterpretation—then the algorithm followed by the machine is immune to sceptical reinterpretation too.

The worry that we faced was whether a particular set of basic terms is up to the job of defining the conditions of correct application for all other terms. The current claim is that *if* one's set of basic terms is up to the job of defining the basic instructions of a universal machine, *then* it is up to the job of defining the application conditions of all other terms. The justification for this claim is as follows. If natural language is computable, then it is computable by any universal computer. The algorithms that such a universal computer follows consist of basic instructions. If those basic instructions are immune to sceptical reinterpretation, then the algorithms are immune to sceptical reinterpretation too. The basic instructions of the universal computer are immune to sceptical reinterpretation if those basic instructions *coincide with our set of basic terms*—the set of terms for which the sceptic's question has already been answered. Therefore, if one's set of basic terms define the basic operations of a universal machine, then that set of basic terms is up to the job of defining the application conditions of every term in natural language.

This is only half of an answer to our worry. How do we know that the antecedent of this conditional is true? How do we know that our set of basic terms *does* define the architecture of a universal computing machine? There are no guarantees, but some encouraging results here. It is well known that the requirements for creating a universal computing machine are minimal. With only a handful of predicates one can specify a universal computing architecture (there is a small industry creating universal computers out of simple and unexpected resources). Existing universal machines include not only those listed above, but also recurrent neural networks (finite-precision inputs/outputs/weights, infinite-precision signals initialized to zero), finite state machines with two stacks, and nearly all programming languages (including URISC, which has only one instruction). Only a modest selection of predicates is needed to create a universal computing machine. It is likely that if an existing response to the sceptic—e.g. Maddy's and McGinn's, Millikan's, or Fodor's—works, then the set of terms for which it works will be adequate to define the architecture of a universal computing machine. If this condition is met, then that set will be able to specify the application conditions of all other terms in natural language.[7]

7. Note that the claim is not that the number of basic terms of a universal computer *must* be small, only that it *can* be small (one could design a universal computer with a labyrinthine architecture that has a huge number of basic operations). How is the number of basic terms counted? The number of basic terms is the minimum number whose meaning needs to be fixed in order to specify the architecture of the computing machine in question. (Suppose you are asked to write out a full specification of the architecture, what is the minimum number of terms you need to use to do so?)

5 Commitment to the CTM

There is a close connection between the basic terms of the algorithm reply and the basic instructions of a universal computing machine. The basic terms of the algorithm reply specify the instructions of a universal computing machine that is able to decide, by following appropriate algorithms, whether a given natural language term correctly applies in a given case. Correspondingly, if whether a natural language term correctly applies in a given case is decidable by a computer, and the basic instructions of that computer are not susceptible to sceptical reinterpretation, then the correct application conditions of natural language are not susceptible to sceptical reinterpretation either. A consequence is that, if one sees the decision as to whether a given term correctly applies as lying within the human mind, a factor influencing the choice of basic terms will be if, and in what sense, one sees the human mind as a computer. If one sees the human mind as computer of a particular sort (classical, connectionist, etc.), then it will be natural to try to specify the rules for applying natural language terms in terms of the basic operations of that machine. This suggests one apparent problem with the computability assumption: it presupposes that the human mind is a computer, and this is far from uncontroversial.

This problem is not as serious as it may seem. The issue need only arise to the extent that one takes Kripke's sceptic as presenting a challenge about individual meaning or mental content. This is one way to read the sceptic's challenge, but not the only way. One could take the sceptic as presenting a challenge about the meaning of terms in public language. On this understanding, an advocate of the computability assumption need not assume that individuals perform computations. The computations that fix non-basic terms could be computations performed by groups. The idea that computations can be performed by groups rather than individuals is not unusual. Hutchins (1995) and Clark (1997) give examples in which groups of humans cooperate and, in conjunction with artefacts, perform a computation that does not take place inside the head of any single individual.[8] If the computations that fix the meaning of non-basic terms are of this public kind, then they need not involve any commitment to individuals performing computations or privately following rules.

By the same reasoning, the computability assumption is compatible with Putnam (1975)'s claim about the division of linguistic labour. The algorithms for correctly applying 'elm' could be computational without lying inside the head of every competent language user; the algorithms could be distributed among experts.[9]

It is worth noting that if one does accept a version of the CTM, then the algorithm reply suggests a possibility that is usually ignored in discussions of Kripke's paradox. This possibility is that the basic terms may themselves not be terms of natural language: they may be basic operations of the human mind. If the human mind is computer, then like any computer, it will have certain basic operations. If the correct application of natural

8. Hutchins (1995) gives the example of computing the position of a US Navy battle group.

9. Putnam (1975)'s point about indexical nature of natural kind terms requires a different treatment. One strategy would be to follow Maddy and McGinn and treat natural kind terms as basic, and hence not requiring algorithms to specify their conditions of application. Another would be to reject Putnam's treatment of natural kinds terms in favour of a descriptivist view; see Mellor (1977), Zymach (1976) for defence of such a view.

language terms is decided within the human mind, then the rules for applying those terms must ultimately be fixed in terms of the basic operations of that computer. Therefore, if one fixes determinacy for those basic operations, then one will thereby fix determinacy for all terms in natural language. Since the basic operations of the human mind are likely to be simple and evolutionarily salient, the task of fixing their determinacy might be easier than that of fixing the determinacy of many natural language terms.

6 Conclusion

Respondents to Kripke's sceptic ought to worry about whether their solutions generalise to the whole of natural language. The algorithm reply provides a way of answering this worry. One might have three concerns about whether the algorithm reply is possible. If the computability assumption and the Church–Turing thesis are correct, then these concerns can be answered. Two further questions remain. The first is whether we, as language users, *in fact* follow the rules given by a particular algorithm reply. For example, in the case of 'plus' it is unlikely that we decide correct application by counting marbles. A full solution would depend on identifying our actual linguistic rules, and this is an empirical matter. The second question is whether a set of basic terms is adequate to define a universal computing machine. The requirements for creating a universal computing machine are minimal, so it is likely that this condition can be met. However, a definitive answer to this question has to wait until particular solutions specify the set of terms for which they work. This is something that remains to be done for many particular solutions. Until these two questions are answered, the algorithm reply remains a promissory note. However, as things stand it is our best chance of solving the auxiliary problem.

Acknowledgements

I would like to thank Peter Lipton and Martin Kusch for comments on an earlier version of this paper.

References

Church, A. 1941. *The Calculi of Lambda-Conversion.* Princeton, NJ: Princeton University Press.

Clark, A. 1997. *Being There.* Cambridge, MA: MIT Press.

Dreyfus, H. L. 1992. *What Computers Still Can't Do.* Cambridge, MA: MIT Press.

Fodor, J. A. 1990. *A Theory of Content and Other Essays.* Cambridge, MA: MIT Press.

Ginet, C. 1992. 'The Dispositionalist Solution to Wittgenstein's Problem About Understanding a Rule: Answering Kripke's Objections'. In *Midwest Studies in Philosophy,* edited by P. A. French, T. E. Jr. Uehling and H. K. Wettstein, 17:53–73. Notre Dame: University of Notre Dame Press.

Haugeland, J. 1981. 'Semantic Engines: An Introduction to Mind Design'. In *Mind Design,* edited by J. Haugeland, 1–34. Cambridge, MA: MIT Press.

Hutchins, E. 1995. *Cognition in the Wild.* Cambridge, MA: MIT Press.

Kripke, S. A. 1982. *Wittgenstein on Rules and Private Language.* Cambridge, MA: MIT Press.

Maddy, P. 1984. 'How the Causal Theorist Follows a Rule'. *Midwest Studies in Philosophy* 9:457–477.

McGinn, C. 1984. *Wittgenstein on Meaning.* Oxford: Blackwell.

Mellor, D. H. 1977. 'Natural Kinds'. *The British Journal for the Philosophy of Science* 28:299–312.

Miller, A., and C. Wright, eds. 2002. *Rule-Following and Meaning.* Chesham: Acumen.

Millikan, R. G. 1990. 'Truth rules, hoverflies, and the Kripke–Wittgenstein paradox'. *Philosophical Review* 99:323–353.

Putnam, H. 1975. 'The Meaning of "Meaning"'. In *Mind, Language and Reality, Philosophical Papers, vol. 2,* 215–271. Cambridge: Cambridge University Press.

Rosch, E. 1973. 'On the Internal Structure of Perceptual and Semantic Categories'. In *Cognitive Development and the Acquisition of Language,* edited by T. Moore. New York, NY: Academic Press.

Searle, J. R. 1980. 'Minds, brains, and programs'. *Behavioral and Brain Sciences* 3:417–424.

Zymach, E. 1976. 'Putnam's Theory of the Reference of Subject Terms'. *The Journal of Philosophy* 73:116–127.